PATHOLOGICAL IMAGE SEGMENTATION FOR NEUROBLASTOMA USING THE GPU

Antonio Ruiz[*], *Jun Kong*[†‡], *Manuel Ujaldon*[*], *Kim Boyer*[†], *Joel Saltz*[‡], *Metin Gurcan*[‡]

[*]Dept. of Computer Architecture, University of Malaga, Malaga (Spain)
[†]Dept. of Electrical and Computer Engineering, Ohio State University, Columbus, OH (USA)
[‡]Dept. of Biomedical Informatics, Ohio State University, Columbus, OH (USA)

ABSTRACT

We present a novel use of GPUs (Graphics Processing Units) for the analysis of histopathological images of neuroblastoma, a childhood cancer. Thanks to the advent of modern microscopy scanners, whole-slide histopathological images can now be acquired but the computational costs to analyze these images using sophisticated image analysis algorithms are usually high. In this study, we have implemented previously developed image analysis algorithms using GPUs to exploit their outstanding processing power and memory bandwidth. The resulting GPU code was contrasted and combined with a C++ implementation on a multicore CPU to maximize parallelism on emerging architectures. Our codes were tested on different classes of images, with performance gain factors about 5.6x when the execution time of a Matlab code running on the CPU is compared with a code running jointly on CPU and GPU.

Index Terms— Neuroblastoma, segmentation, differentiation grading, computer-aided prognosis, GPU.

1. INTRODUCTION

Neuroblastoma (NB) is an embryonal tumor usually originating from the sympathetic nervous system [1]. Due to the large variation in its morphological structure, the prognosis of this disease is challenging and it affects the treatment plan. In current clinical practice, neuroblastoma classification is carried out by highly trained pathologists with visual examinations of pathological slides under the microscope according to the International Neuroblastoma Classification System developed by Shimada et al.

Visual evaluation of histopathological slides may lead to large intra- and inter-reader variability. A recent study reports that there is a 20% discrepancy between central and institutional reviewers for neuroblastoma prognosis [2]. Since it is now possible to digitize neuroblastoma slides, these digital images can be analyzed using the computer. The quantitative information provided by the computer analysis may lead to better reproducibility. Additionally, the computer can analyze every location in each slide and for every slide of the tumor, therefore reducing the chances of mis-prognosis due to visual sampling of heterogenous tumors.

We are developing computer-assisted prognosis (CAP) system for neuroblastoma (NB-CAP). This system analyzes the content of neuroblastoma slides and quantifies the morphological features used in neuroblastoma prognosis. A major component of this system involves the analysis of grade of differentiation. We have developed algorithms to determine the percentages of grade of differentiation in a given slide as undifferenting, poorly-differentiating and differentiated [3]. These percentages can then be used in the decision tree of the prognosis system.

One of the key steps in NB-CAP system is segmentation of basic histological components for grade of differentiation analysis. In our previous study, we developed a novel segmentation algorithm called EMLDA [3]. Due to large sizes of the neuroblastoma slides and the computational demands of the algorithm, it takes a long time to run. Therefore, we explored a multi-resolution approach and feature selection algorithms to reduce the execution times [4]. In the multi-resolution approach, the image is processed by dividing it into tiles and each tile is analyzed at the lowest resolution first. If the processing results are satisfactory, the execution stops. Otherwise, the image is processed at the next higher resolution. Not all the features are necessary at different resolutions and an automated feature selection algorithm is employed to select the most discriminating features. Fewer features require less time to calculate.

In addition to the algorithmic approaches to reduce the computational demands, novel architectures can be used to speed up the execution times. Our work explores the power of GPUs and combine them with multicore CPUs to accelerate the processing of histopathological images. Data bandwidth and arithmetic intensity is fully exploited in our graphics implementation and different ways of parallelism are enabled in our combined (GPU and CPU) implementation to determine the optimal execution.

This work was partially supported by the US National Science Foundation (#CNS-0643969, #CNS-0403342, #CNS-0615155, #CCF-0342615), by the NIH NIBIB BISTI (#P20EB000591), NCI caBIG core middleware development (79077CBS10), Ministry of Education of Spain (PR-2007-0014) and Junta de Andalucía of Spain (P06-TIC-02109). The authors would like to thank Dr. Hiroyuki Shimada for providing the neuroblastoma images, for guidance and helpful discussions.

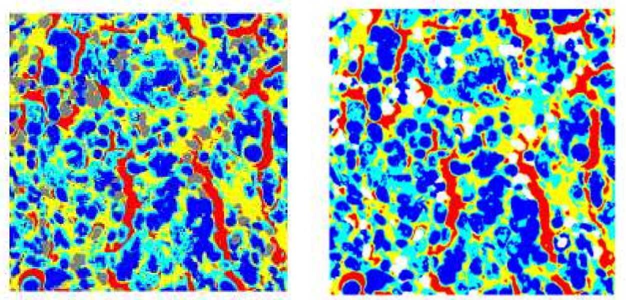

(a) EMLDA segmented. (b) K-means segmented.

Fig. 1. Outputs of (a) EMLDA and (b) K-means segmentation algorithms for an undifferentiated neuroblastoma image.

2. COMPUTERIZED SEGMENTATION ON NEUROBLASTOMA IMAGES

In this study, the input images are produced using a digital scanner (ScanScope T2 digitizer, Aperio, San Diego, CA) that digitizes neuroblastoma biopsies sliced at 5 μm and dyed with haematoxylin and eosin (H&E) stains. For each H&E image, four cytological components, namely nuclei, cytoplasm, neuropil and inter-cellular white space are to be identified. Two segmentation methods, EMLDA and K-means, are applied to H&E images for partitioning different components in our investigations.

2.1. Statistical feature construction

Cytological components in an H&E image typically exhibit distinct colors, which can be used for their characterization. For instance, nuclei regions usually appear as dark blue color; blue-purple colors often indicate the cytoplasm regions, and neuropil components are, in most cases, characterized by pink. In addition to the color information, the local color change associated with each tissue structure also provides a rich source of discriminative information to segment images. As a result, a feature vector to represent and identify different cytological components is established using both information extracted from color and texture analysis. This vector contains a total of six elements. The first three components are the L, A^* and B^* color channels, as the LA^*B^* color space preserves the color perceptual uniformity, and the last three components are the texture counterpart consisting of entropy statistics computed with a 9×9 window shifting across the R, G, and B color channels.

2.2. Segmentation algorithms

2.2.1. EMLDA

We have developed a novel segmentation algorithm, called EMLDA (Expectation-Maximization Linear Discriminant Analysis) [3]. The EMLDA fuses the Fisher-Rao criterion into the generic Expectation-Maximization algorithm. On each iteration, it maximizes the Fisher-Rao criterion and estimates the parametric distributions. Let us denote $X = \{x | x \in \mathcal{R}^p\}$ as the data set consisting of C classes in the feature space, where p is the dimension of the feature space. The cost function to be maximized can then be described as follows:

$$J(V|\theta) = \frac{|V^T S_B(\theta) V|}{|V^T S_W(\theta) V|} \tag{1}$$

where V is a $p \times s$ matrix where each column is a discriminant vector, $s \leq C - 1$; θ is the labeling configuration coming from the previous step, and S_B and S_W are the between- and within-class scatter matrices that are symmetric and positive-definite [5].

Each feature data x is then mapped to a correspondence $\widetilde{x}$ in a lower dimensional subspace of R^p using the discriminant vectors from the matrix $V^*(\theta)$ that maximizes $J(V|\theta)$.

$$\widetilde{X} = \{\widetilde{x} | \widetilde{x} \in R^s\} \text{ and } \widetilde{x} = \left(V^*(\theta)\right)^T x \tag{2}$$

where the new subspace spanned by $\{v_1^*(\theta), v_2^*(\theta), \cdots, v_s^*(\theta)\}$, *i.e.* columns of $V^*(\theta)$, is the subspace where the data $\widetilde{X} = \{\widetilde{x} | \widetilde{x} \in R^s\}$ can be best discriminated in terms of the Fisher-Rao criterion.

Next, we update the label configuration from θ to $\widetilde{\theta}$ in terms of the rule that the updated label of each data in the reduced dimensional space is the one associated with the class having the shortest distance from its centroid.

The above two steps are then iteratively conducted until $J(V|\theta)$ starts decreasing after a minimum number of iterations (10 in our application). To initialize this iterative process, we obtain the initial label configuration θ_0 by the classical K-means clustering method.

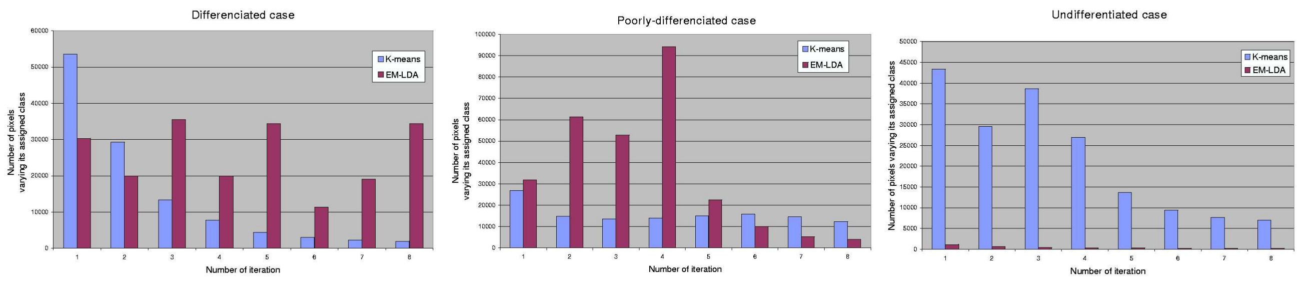
2.2.2. K-means

In our previous work, we compared the processing times by one of the standard segmentation algorithms: K-means.
K-means is an unsupervised method where an iterative process aims at minimizing a cost function:

$$J(\theta) = \sum_{i=1}^{C} \sum_{j=1}^{n_i(\theta)} \|x_{ij} - m_i(\theta)\|^2 \tag{3}$$

where C is the number of components to be identified, θ is the labelling configuration, x_{ij} is the j^{th} feature data ($j \in \{1, 2, \cdots, n_i(\theta)\}$) in the i^{th} component, and $m_i(\theta)$ is the centroid of the class i.

By repeating the assignment of feature data to classes having the closest centroids and the estimation of class-conditional centroids, we can approach the local minimum of the cost function. When $J(\theta)$ converges to the minimum,



(a) Differentiated. (b) Poorly-differentiated. (c) Undifferentiated.

Fig. 2. Evolution of the clustering process during the first eight iterations for each of our segmentation methods under different types of input images: (a) Differentiated. (b) Poorly-differentiated. (c) Undifferentiated.

each group of the image pixels having one distinct class label represents one segmented object component.

2.3. Convergence speed

In order to assess the computational cost of each of the two segmentation methods, we evaluated their convergence speed for the three input images with a different grade of neuroblastic differentiation: differentiated, poorly-differentiated, and undifferentiated. Figure 2 outlines the behavior of each method during the first eight iterations. We selected the number of pixels changing from its previously assigned class as a rough estimate of the stability of the classifying process as well as its convergence speed. Figure 2 shows that the undifferentiated class requires a less number of iterations using K-means and the other two classes are more in favor of the EMLDA.

When comparable final recognition accuracies are achieved on 129 test images, the mean and standard deviation of the computational cost associated with the EMLDA are 13.6 and 4.93 seconds, whereas for the K-means are 29.6 and 9.92 seconds, respectively [3].

3. GPU IMPLEMENTATION

We implemented both the EMLDA and K-means based segmentation methods using CUDA so that a comparison in running times in GPU can be made. CUDA [6] is a high-level programming interface consisting of a set of library functions which can be coded as an extension to the C language. A compiler generates executable code for the GPU, which is treated by the CPU as a multicore co-processor.

For the programmer, the CUDA model is a collection of threads running in parallel which divide the hardware resources equally among themselves, with each thread and block having a unique ID accessed during its execution to process different sets of data in a SIMD (Single Instruction Multiple Data) fashion.

In CUDA, all threads can access any memory location, but performance boosts with the use of shared memory, whenever data to be collectively read belong to different memory banks. This way, the CUDA design does not suffer from constraints when accessing memory, though the access times vary for different types of memory.

Algorithms with higher number of conditional statements slow down the GPU implementation. This is because the GPU is a platform aimed for a streaming execution model where data require to be independent from each other, and, preferably, on a single data flow. Although the C++ implementation of the EMLDA algorithm per iteration is faster than the K-means algorithm for the specific problem we are studying, the CUDA implementation reversed the execution time order.

We can distinguish two parts of the EMLDA algorithm from the GPU implementation viewpoint. The expectation-maximization evaluation requires many data dependencies and conditional statements. Most of these could not be optimized on the GPU, hence they need to be executed on the CPU. Then, the LDA part (Eq. 2) consists mostly of matrix algebra operations, which are more favorable to a SIMD type processing, in particular, to the GPU's ability to simultaneously process almost one hundred operations on concurrent pixel processors. The most computationally demanding part of the EMLDA algorithm was the computation of the generalized version of the eigenvectors required for LDA.

The K-means algorithm requires few conditional statements and the most time consuming step is the distance operator. This operator is very common to graphics processing, which makes it straightforward to be mapped onto the GPU, resulting in fast execution. Since it contains less number of conditional statements, K-means can perform more bulk processing. Therefore, this algorithm can take advantage of memory bandwidth more generously and results in less computational cost for a single iteration on the GPU.

4. EXPERIMENTAL RESULTS

To demonstrate the effectiveness of our computational optimization using the GPU, we have conducted a number of experiments on a state-of-the-art computer equipped with Nvidia GeForce 8800 GPU at 575 MHz and a Intel Core 2 Duo CPU running at 2.13 GHz.

	Image size	CPU execution time using C++	Time per pixel (in nanoseconds)
	59412×64990	2485 secs.	643.58 ns.
	53614×103754	10118 secs.	1818.55 ns.
	68443×75607	6081 secs.	1175.12 ns.
Average	60490×81450	6228 secs.	1212.35 ns.

Table 1. Execution times for the segmentation on the CPU on different input images using C++ and our EMLDA clustering method. The average time spent is one hour and 45 minutes.

Clustering method	C++ execution time (CPU)	CUDA execution time (GPU)	Speed-up on GPU
EMLDA	174 msc.	120 msc.	1.45x
K-means	220 msc.	64 msc.	3.43x

Table 2. Execution times in milliseconds per iteration on each hardware platform for a single 512x512 image tile.

Table 1 lists the calculation times for three sample images when implemented in C++ running on CPU. The average size of the image is 60K×81K, which takes several hours to run in Matlab. This running time was reduced to one hour and 45 minutes on the CPU by implementing the methods in C++. The C++ implementation also provides a more realistic comparison with our CUDA code on the GPU.

Table 2 shows the execution times on both platforms for a single iteration on a 512x512 image tile. As anticipated, K-means achieves a better speed-up factor per iteration of 3.43x on the GPU, whereas the EMLDA benefits from a modest 1.45x acceleration. Consequently, the EMLDA becomes twice slower than the K-means on the GPU and slightly faster on the CPU. However, the EMLDA requires less number of iterations than K-means on average when run on NB images, that is, it is 2.13 times faster in the convergence process.

We may select both methods to run on the GPU, but that will result in an idle CPU. The best approach consists of enabling a bi-processor platform to engage the GPU contribution, and then process a proportional number of tiles on each processor to maximize parallelism depending on the segmentation method. This requires a marginal participation of the CPU acting as host of the GPU code for shipping data; for this purpose, the second CPU core is enabled and communications are handled using asynchronous capabilities of CUDA.

For a 53614x103754 input image, a total of 13727 tiles of 512x512 pixels were processed after discarding those tiles determined as background. When using EMLDA, the CPU segments 5602 tiles and the GPU processes 8125 tiles for a total time of 975 seconds per iteration on each side. When selecting K-means, the CPU takes 3093 tiles and the GPU 10634, leading to an overall time of 680.5 seconds. Since the total execution time for the input image is 5465 seconds per iteration using Matlab and EMLDA, we achieve an overall speed-up factor of 5.6x when enabling the CPU-GPU combined execution outlined throughout this paper. Also, an additional 1.43x improvement factor would be attained for those cases in which the K-means segmentation method would require a similar number of iterations to converge as EMLDA.

5. CONCLUSIONS

In this paper we present a novel computational architecture to process histopathological images. The implementation details are illustrated through a computer-aided prognosis application for neuroblastoma (NB-CAP). After implementing two alternative forms of a time-consuming step in NB-CAP, we analyzed the trade-offs for an efficient implementation of each segmentation method. The implementation was done by translating the original Matlab code into a compiled C++ version, and then implementing the same algorithm on a GPU using CUDA. This implementation revealed that one particular segmentation algorithm (EMLDA) is more appropriate to be implemented using the CPU architecture, whereas the other algorithm (K-means) has better properties for a streaming execution on the GPU.

The GPU is particularly suitable for pathological image analysis because it provides streaming programming and huge memory bandwidth for very large scale input images. GPUs are highly scalable and evolve towards general-purpose architectures [7], where we envision biomedical image processing as one of the most exciting fields to benefit from subsequent developments.

6. REFERENCES

[1] F. Alexander, "Neuroblastoma," *The Urologic Clinics of North America*, vol. 27, pp. 383–392, 2000.

[2] L.A. Teot, R.S.A. Khayat, S. Qualman, G. Reaman, and D. Parham, "The problems and promise of central pathology review: Development of a standardized procedure for the children's oncology group," *Pediatric and Developmental Pathology*, vol. 10, pp. 199–207, 2007.

[3] J. Kong, H. Shimada, K. Boyer, J. Saltz, and M. Gurcan, "Image analysis for automated assessment of grade of neuroblastic differentiation," in *Intl. Symposium on Biomedical Imaging*, 2007, pp. 61–64.

[4] J. Kong, O. Sertel, H. Shimada, K. Boyer, J. Saltz, and M. Gurcan, "Computer-aided grading of neuroblastic differentiation: Multi-resolution and multi-classifier approach," in *Proceedings IEEE ICIP 2007*.

[5] G.J. McLachlan, "Discriminant analysis and statistical pattern recognition," Wiley-Interscience, 1992.

[6] CUDA, http://developer.nvidia.com/object/cuda.html.

[7] GPGPU, A web site dedicated to the general-purpose on the GPU: http://www.gpgpu.org.